# Simpler is Better: Finding the Best Reward Function in Long Chain-of-Thought Reinforcement Learning for Small Language Models

Luning Wang[*1] Zichen Zhang[*1] Junkuan Liu[*1]

## 1. Introduction

Large Language Models (LLMs) have been evolving rapidly in recent years. A critical technology that contributes to this development is Reinforcement Learning (RL). Recently, strong LLMs like DeepSeek-R1 (DeepSeek-AI et al., 2025), which show very promising performance on complex reasoning tasks such as solving math and coding problems, have adopted verifiable rules-based rewards. Rule-based rewards are pre-defined, so they are more interpretable and controllable by developers and could be adapted for different reasoning tasks. Subsequent paper (Yeo et al., 2025) further discuss the mechanics of long CoT reasoning, and propose to use reward shaping to stabilize and control CoT length while improving accuracy. Specifically, they propose cosine reward with a repetition penalty, which could stabilize CoT growth while encouraging emergent reasoning behaviors such as branching and backtracking. However, there lacks the discussion on the effect of those different types of rewards on small language models (SLMs), which typically have a parameter size <7B and could behave differently from those large models. Therefore, We focus on the effect of different rewards on small models ($\sim$3B). We first propose a dynamic reward that extends the concept of cosine reward, and then experiment several kinds of rewards (normal, cosine, dynamic) on the chosen SLMs. With careful observation and analysis, we provide several key insights which could benefit future studies in this field . Code is available at `https://github.com/zichenzhang04/r1-dynamic-penalty`.

## 2. Methods

Our approach builds on the Group Relative Policy Optimization (GRPO) (Shao et al., 2024) framework for reinforcement learning, utilizing the Unsloth library (Daniel Han & team, 2023) to efficiently fine-tune language models with verifiable, rule-based reward functions. We focus on aligning model behavior through rewards that incentivize correctness, formatting, brevity, and non-redundancy in Chain-of-Thought (CoT) reasoning (Wei et al., 2022).

We first replicate and compare two types of reward configurations: the **Normal (Classic) Reward** (DeepSeek-AI et al., 2025) and the **Cosine Reward** (Yeo et al., 2025), each composed of several sub-reward functions. These are designed to encourage adaptive inference depth and prevent reward hacking through repetition. We further propose a new **Dynamic Reward** to further stabilize the training, and compare its effect with the previous two rewards.

### 2.1. Framework

We use the Unsloth library to load and LoRA-adapt (Hu et al., 2022) a base instruction-tuned model, Qwen2.5-3B (Team, 2024), into a GRPO-compatible training setup. The model is trained on GSM8K (Cobbe et al., 2021), with optional evaluation on MATH500 (Lightman et al., 2023) and GSM8K's test split. A set of modular, rule-based reward functions is defined and passed to the `GRPOTrainer` in Unsloth, which uses the cumulative reward signal to compute policy advantages and drive the optimization. The training loop is further customized to correctly handle inference logging and reward normalization per training group.

### 2.2. Correctness Reward

#### 2.2.1. Naive Correctness Reward

This function compares the extracted answer from the model's output with the ground-truth answer using either string equality or symbolic equivalence. It returns a high reward (+2.0) for correct answers and zero otherwise. During training, it also logs accuracy metrics and "aha moment" keywords (e.g., wait, recheck) for interoperability.

#### 2.2.2. Cosine Reward

The Naive Correctness Reward often leads to unstable training. To regulate CoT length and avoid unnecessary verbosity, we use a Cosine Scaled Sparse Reward strategy introduced in Yeo et al. 2025 that dynamically adjusts reward values based on completion length:

For **correct** answers, shorter completions are preferred, with rewards decreasing from $+2.0$ (shortest) to $+1.0$ (longest).

For **incorrect** answers, longer completions are less punished, with rewards increasing from $-10.0$ (shortest) to $0.0$ (longest).

This cosine schedule encourages the model to use computation efficiently—thinking longer when unsure, but stopping early when confident.

## 2.3. Format Rewards

Following Yeo et al. 2025, we add two functions to verify that the output adheres to the expected XML-style CoT structure:

**Strict Format Reward**: Requires precise formatting, with each tag (e.g., `<reasoning>`, `<answer>`) on its own line and closed correctly. Returns $+1.0$ if format is correct, $0.0$ otherwise.

**Soft Format Reward**: Loosens constraints to check *only* the presence of `<reasoning>` and `<answer>` tags, allowing *arbitrary* spacing. Returns $+1.0$ for valid presence, $-2.0$ otherwise.

Additionally, the **XML Count Reward** returns a fractional reward (in $0.125$ increments) based on the presence and ordering of XML tags, helping shape early-stage behavior.

## 2.4. Integer Reward

As used in Yeo et al. 2025, this reward encourages the model to produce an integer-type final answer, given that the training dataset GSM8K contains integer answers. If the extracted answer is a digit, it receives a small reward ($+0.5$), otherwise a large penalty ($-10.0$). This supports early-stage correctness shaping, even before the model learns full CoT reasoning.

## 2.5. Repetition Reward

To discourage reward hacking via n-gram repetition, we include a dense repetition penalty function (Yeo et al., 2025). It penalizes completions with repeated n-grams (default $n = 20$) by up to $-1.0$ for each repeated segment. This reward is automatically included in the cosine reward variant to prevent degenerate repetition loops that artificially increase length-based reward.

## 2.6. Dynamic Reward

At the start of the training, when accuracy is low, the Cosine Reward, by definition, incentivizes the model to think longer. This leads to increased reward hacking through repetition, demanding a stronger repetition penalty. In the middle and the end of the training, when accuracy becomes higher, reward hacking becomes less likely, requiring a weaker repetition penalty. Therefore, we propose to dynamically adjust the weight of repetition penalty, where the reward function is given as:

$$R_{\text{dynamic}} = (1-w_{\text{repetition}})R_{\text{cosine}} - w_{\text{repetition}} \sum_{t=1}^{T} P_{\text{repetition}}(x_t) \tag{1}$$

Here, $R_{\text{cosine}}$ is the sparse Cosine Reward(Yeo et al., 2025) applied to the entire reasoning trajectory, designed to both enforce correctness, format and regulate the CoT length. $P_{\text{repetition}}(x_t)$ is a dense penalty(Yeo et al., 2025) that applies to each repeated token $x_t$ to mitigate reward hacking. The dynamic weight $w_{\text{repetition}} \in [0, 1]$ dynamically adjusts between training stages, balancing reasoning depth and repetition suppression. It is defined as:

$$w_{\text{repetition}} = \sigma(\alpha f_{\text{rep}}(X) + \beta), \tag{2}$$

where $f_{\text{rep}}(X)$ quantifies the repetition frequency in the generated sequence $X$, $\sigma(\cdot)$ is the sigmoid function that ensures a smooth transition, and $\alpha, \beta$ are tunable hyperparameters controlling the rate of adaptation. This formulation enables dynamic scaling of the repetition penalty, penalizing excessive repetition during early training while reducing its effect as model accuracy improves.

## 3. Experiments

### 3.1. Settings

#### 3.1.1. Model and Dataset

In our experiments, we choose Qwen2.5-3B-Instruct (Team, 2024) as our base model, which is different from the paper (Yeo et al., 2025) that we replicate due to our limit computation resources. We train the model on GSM8K (Cobbe et al., 2021), and evaluate on MATH500 and GSM8K's test split during training. Due to the limit of time, we utilize the first 256 samples from MATH500 and GSM8K's test split for evaluation. These two datasets could both be regarded as in-domain evaluation for the model as they share similar requirements of capability with our training set.

#### 3.1.2. Training Hyperparameters

The training hyperparameters are mainly set according to the default configuration of the framework, with learning_rate=5e-6, adam_beta1=0.9, adam_beta2=0.99, weight_decay=0.1 and warmup_ratio=0.1. We also set the maximum training steps as 500 and group size as 8 during the GRPO optimization. As for the sequence length limit, we set max_prompt_length = 1024 with max_completion_length = 1024. We do evaluation on the chosen subset every 25 steps during training.

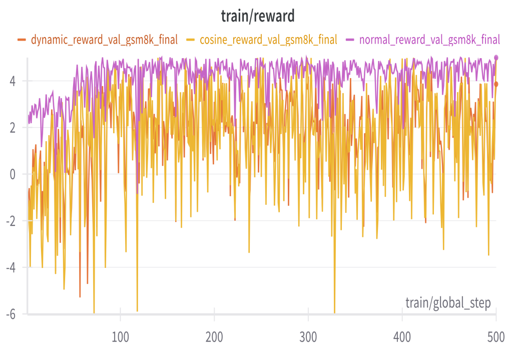
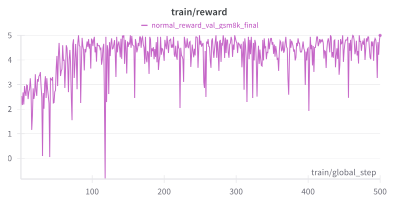
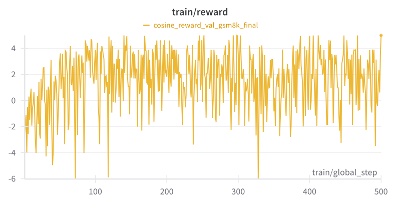


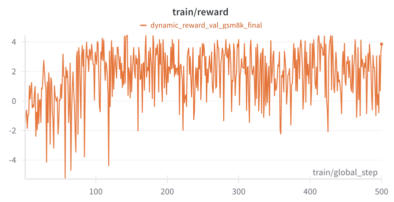
*Figure 1.* Rewards of Correctness, Cosine and Dynamic



(a) Normal Reward



(b) Cosine Reward



(c) Dynamic Reward

*Figure 2.* Training rewards under normal, cosine, and dynamic reward schemes

## 3.2. Results of different rewards

### 3.2.1. Correctness Reward and Cosine Reward

Figure 1 presents the total rewards under the normal reward, the cosine reward, and dynamic reward scheme. As expected, the rewards under the cosine scheme and dynamic scheme are consistently lower, reflecting its stricter evaluation criteria compared to the normal reward. More specifically, Figure 2 shows three kinds of reward individually, further illustrating the more stringent nature of the cosine and dynamic metric. Since there are only 8 samples evaluated at each global step, the resulting curves exhibit considerable fluctuation and offer limited insight into the model's overall performance. Therefore, we will refer to the evaluation results discussed in the following sections to better assess the performance trends.

### 3.2.2. Format Reward

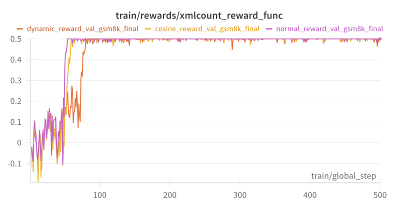
Figure 3(a) shows the XML count reward during training, which increases rapidly and saturates within the first $\sim$50 training steps. Figure 3(b) and Figure 3(c) present the strict format reward and the soft format reward, respectively. We observe that the soft format reward remains consistently high throughout most steps, whereas the strict format reward stays low initially and begins to rise sharply between steps 50 and 100. These results suggest that the model first learns to correctly output XML tags and subsequently focuses on adhering to the strict formatting requirements, eventually reaching saturation in formatting rewards. This progression highlights the role of the XML count reward in shaping the model's early-stage behavior. Moreover, the results indicate that formatting is a relatively easy challenge for the selected model, as it internalizes the desired structure quite rapidly.

### 3.2.3. Integer Reward

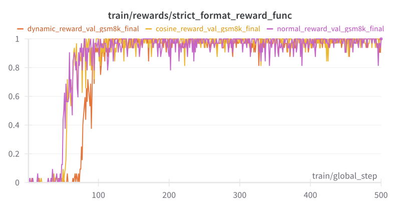
Figure 4 illustrates the integer reward throughout training. We do not observe any clear trend in the reward's progression. Through case studies, we find that some non-integer extracted answers are caused by truncated responses — the model is often still reasoning when it is forced to stop, leading to the failure to output a properly formatted final answer. The maximum generation length was set to a relatively small value (1024) to ensure efficient training given our limited time and computational resources. As a result, the regularization effect of the integer reward remains unclear, and additional experiments would be necessary for a more thorough investigation.

### 3.2.4. Repetition Reward

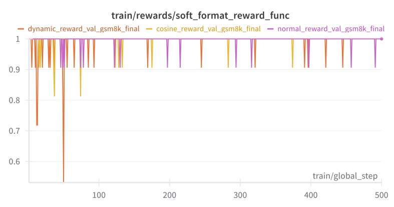
Figure 5 shows the repetition reward during training. We observe that repetition cases are very rare in our experiments, resulting in this reward having almost no impact. This could be attributed to the intrinsic characteristics of our model,



(a) xml count reward



(b) strict format Reward



(c) soft format Reward

*Figure 3.* Format rewards

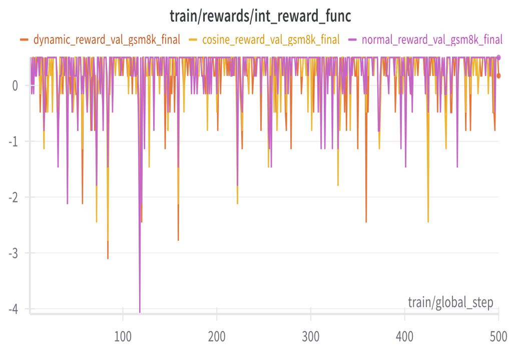


*Figure 4.* Integer Rewards

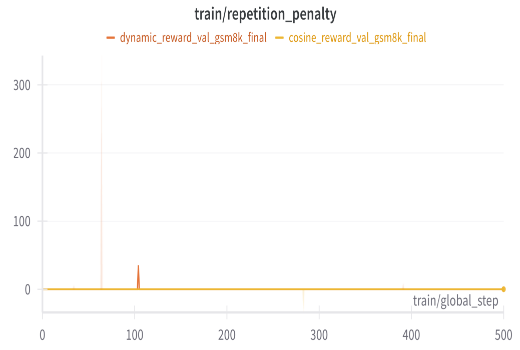


*Figure 5.* Repetition Rewards

though further investigation would be needed for deeper insights.

### 3.3. Results of completion and reasoning length

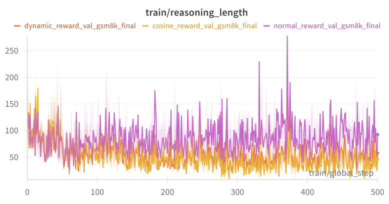
Figure 6(a) shows the reasoning lengths under three kinds of reward scheme, while Figure 6(b) presents their corresponding completion lengths. We observe that under the normal reward, both reasoning and completion lengths exhibit a slight upward trend. In contrast, under the cosine reward and dynamic reward, both lengths are significantly suppressed and eventually stabilize at a relatively low level. This highlights the length regularization effect of the cosine reward, which discourages the model from producing overly long reasoning processes while still aiming to maintain the correctness of the final answer. Essentially, the cosine reward seeks to strike a balance between reasoning length and answer accuracy, recognizing that longer reasoning typically improves correctness, but at the cost of verbosity.

### 3.4. Results of special words

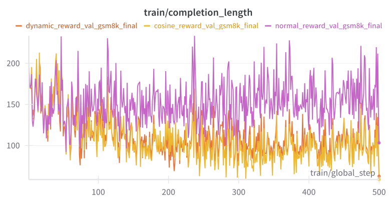
We also investigate the occurrence of certain special words that signal the model's self-reflection, commonly referred to as ”aha” words in Deepseek-R1's technical report (DeepSeek-AI et al., 2025). These ”aha” words are considered important markers of complex reasoning and serve as useful indicators of a model's reasoning ability. Specifically, we track the occurrence of five selected ”aha” words — ”alternatively”, ”wait”, ”retry”, ”recheck”, and ”however” — based on prior studies (Yeo et al., 2025). Figure 7(a) presents the overall occurrence across all ”aha” words, while Figure 7(b) $\sim$ Figure 7(f) show the breakdown for each word individually. We observe that the model most frequently produces ”however”, with occasional appearances of ”wait” and ”recheck”, and almost no instances of ”alternatively” or ”retry”. No clear trend is observed in the frequency of ”aha” words over the course of training, which might be attributed to the model's limited reasoning capabilities and warrants further investigation.

### 3.5. Results of evaluation

To monitor the change in model performance during training, we conduct evaluations every 25 steps on our selected subsets. Figure 8(a) shows the results on MATH500, while Figure 8(b) shows the results on the GSM8K test set. We observe that on both datasets, the model's performance improves steadily under the normal reward scheme, but drops and fluctuates when trained with cosine and dynamic rewards. This outcome contradicts previous findings (Yeo et al., 2025), which report that cosine rewards generally boost model performance while suppressing reasoning length. We suspect that the discrepancy stems from differences between the models: their reported gains were

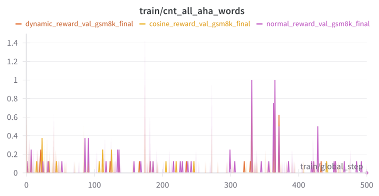


(a) Reasoning Length

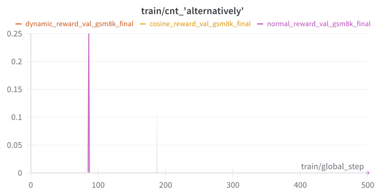


(b) Completion Length

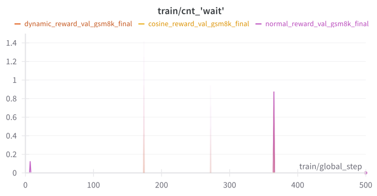
*Figure 6.* Sequence Length



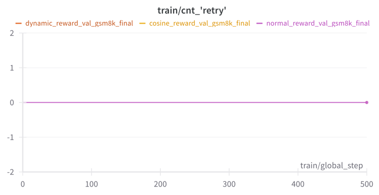
(a) All Aha Words



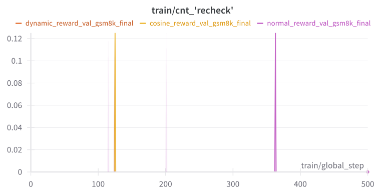
(b) Alternatively



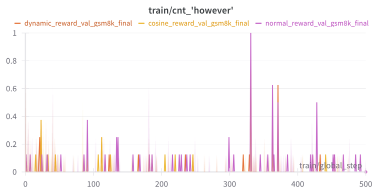
(c) Wait



(d) Retry



(e) Recheck



(f) However

*Figure 7.* Occurrence of ’Aha’ Words during training.

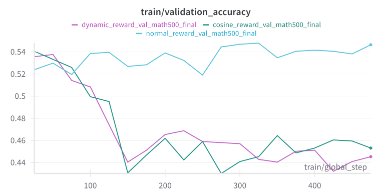
based on larger models, such as Llama-3.1-8B (Touvron et al., 2023) and Qwen2.5-7B-Math (Team, 2024), both with around twice as many parameters as our Qwen2.5-3B-Instruct model. Under this assumption, the redundancy of overly long reasoning identified in larger models may not exist in our smaller model — in fact, a smaller model might still benefit from longer reasoning steps to produce accurate answers. These initial results suggest that the cosine reward may not be directly suitable for our smaller model, and further investigation is needed to find better strategies.

## 4. Preliminary Conclusions and Next Steps

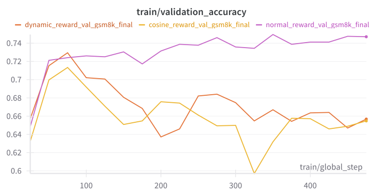
**Preliminary Conclusions.** Our initial experiments show that the Normal (Classic) Reward scheme steadily improves model performance on the chosen subsets, whereas the Cosine Reward scheme and Dynamic Reward scheme lead to noticeable fluctuations and an overall drop in performance. This suggests that length-based penalties — as introduced by the Cosine Reward — might actually hurt smaller models (around 3B parameters) by discouraging them from producing sufficiently detailed reasoning, which is often necessary for accurate answers. This finding contrasts with results reported for larger models ($\geq$7B parameters), where reducing reasoning length helped without sacrificing accuracy. That said, we do find that format-based rewards are quite effective. They help the model quickly learn to follow XML-style output, with the corresponding rewards saturating early as the model internalizes the formatting rules.

On the other hand, the Integer Reward had limited effect, likely because the relatively small maximum sequence length (1024 tokens) led to truncation before the final numeric answers could be fully generated. Similarly, the Repetition Reward was rarely triggered, suggesting that our model naturally avoids repetitive outputs. Altogether, these results point to a broader observation: smaller models may require more careful or customized reward shaping to properly balance correctness and reasoning brevity.

**Next Steps.** Moving forward, we plan to start with a more extensive hyperparameter exploration. By tuning the reward coefficients and — if resources allow — increasing the maximum generation length, we hope to find a better trade-



(a) Accuracy in MATH500



(b) Accuracy in GSM8K

*Figure 8.* Validation Accuracy

off between keeping answers concise and ensuring high accuracy.

At the same time, we will dive deeper into analyzing failure cases, especially focusing on instances where truncated reasoning steps prevent the model from reaching a valid final answer. A closer look here should help us refine training strategies and reduce the frequency of generation cutoffs that hurt accuracy.

Additionally, we aim to improve reward shaping itself. Some ideas include shifting more weight toward correctness, softening the penalty curve for the Cosine Reward when reasoning chains get longer, or introducing partial-credit scoring to give more fine-grained feedback during training.

Finally, we plan to broaden our evaluation beyond GSM8K and MATH500, testing across a wider range of math and reasoning benchmarks. The goal is to make sure any improvements we find in reward design aren't just dataset-specific, but generalize well across different kinds of problems.

## 5. Contribution of team members

Here are the contributions of each team member:

- Zichen Zhang: Propose the core idea of our project, and implement the training framework with all the three types of rewards.
- Luning Wang: Implement the evaluation code for validation on GSM8K. Conduct most experiments on normal reward and cosine reward.
- Junkuan Liu: Implement the evaluation code for validation on TheoremQA. Conduct most experiments on dynamic reward.

Everyone contributes to the proposal report, progress report, presentation slide and final report. Overall, each team member contributes equally to the project.

## References

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Daniel Han, M. H. and team, U. Unsloth, 2023. URL http://github.com/unslothai/unsloth.

DeepSeek-AI, Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., Zhang, X., Yu, X., Wu, Y., Wu, Z. F., Gou, Z., Shao, Z., Li, Z., Gao, Z., Liu, A., Xue, B., Wang, B., Wu, B., Feng, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., Dai, D., Chen, D., Ji, D., Li, E., Lin, F., Dai, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Bao, H., Xu, H., Wang, H., Ding, H., Xin, H., Gao, H., Qu, H., Li, H., Guo, J., Li, J., Wang, J., Chen, J., Yuan, J., Qiu, J., Li, J., Cai, J. L., Ni, J., Liang, J., Chen, J., Dong, K., Hu, K., Gao, K., Guan, K., Huang, K., Yu, K., Wang, L., Zhang, L., Zhao, L., Wang, L., Zhang, L., Xu, L., Xia, L., Zhang, M., Zhang, M., Tang, M., Li, M., Wang, M., Li, M., Tian, N., Huang, P., Zhang, P., Wang, Q., Chen, Q., Du, Q., Ge, R., Zhang, R., Pan, R., Wang, R., Chen, R. J., Jin, R. L., Chen, R., Lu, S., Zhou, S., Chen, S., Ye, S., Wang, S., Yu, S., Zhou, S., Pan, S., Li, S. S., Zhou, S., Wu, S., Ye, S., Yun, T., Pei, T., Sun, T., Wang, T., Zeng, W., Zhao, W., Liu, W., Liang, W., Gao, W., Yu, W., Zhang, W., Xiao, W. L., An, W., Liu, X., Wang, X., Chen, X., Nie, X., Cheng, X., Liu, X., Xie, X., Liu, X., Yang, X., Li, X., Su, X., Lin, X., Li, X. Q., Jin, X., Shen, X., Chen, X., Sun, X., Wang, X., Song, X., Zhou, X., Wang, X., Shan, X., Li, Y. K., Wang, Y. Q., Wei, Y. X., Zhang, Y., Xu, Y., Li, Y., Zhao, Y., Sun, Y., Wang, Y., Yu, Y., Zhang, Y., Shi, Y., Xiong, Y., He, Y., Piao, Y., Wang, Y., Tan, Y., Ma, Y., Liu, Y., Guo, Y., Ou, Y., Wang, Y., Gong, Y., Zou, Y., He, Y., Xiong, Y., Luo, Y., You, Y., Liu, Y., Zhou, Y., Zhu, Y. X., Xu, Y., Huang, Y., Li, Y., Zheng, Y., Zhu, Y., Ma, Y., Tang, Y., Zha, Y., Yan, Y., Ren, Z. Z., Ren, Z., Sha, Z., Fu, Z., Xu, Z., Xie, Z., Zhang, Z., Hao, Z., Ma, Z., Yan, Z., Wu, Z., Gu, Z., Zhu, Z., Liu, Z., Li, Z., Xie, Z., Song, Z., Pan, Z., Huang, Z., Xu, Z., Zhang, Z., and Zhang, Z. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y. K., Wu, Y., and Guo, D. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Team, Q. Qwen2.5: A party of foundation models, September 2024. URL https://qwenlm.github.io/blog/qwen2.5/.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E. H., Le, Q. V., and Zhou, D. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Yeo, E., Tong, Y., Niu, M., Neubig, G., and Yue, X. Demystifying long chain-of-thought reasoning in llms, 2025. URL https://arxiv.org/abs/2502.03373.